GMBI: Decoupled Polynomial State-Space Iterations for Matrix Polar Approximation, with an Application to Muon

Anonymous authors
Paper under double-blind review

Abstract

Matrix polar approximation is a recurring primitive in orthogonalization and low-precision optimization. We introduce Gram-matrix-based iterations (GMBI), a decoupled polynomial state-space parameterization that separately controls propagation of a square auxiliary state and accumulation of the rectangular output under a restricted matrix-computation budget. Within the Polar Express (PE) computational framework, GMBI supports both compact short graphs and robust full-restart graphs. We fit restart graphs phasewise: a guarded two-layer prefix is frozen and run on calibration matrices in FP16, and its output is materialized to rebuild true Gram states. The suffix is optimized by scalar rollouts from these realized restart spectra, adapting it to prefix-induced distribution shifts and finite-precision effects. In controlled FP16 experiments, Basic GMBI reduces singular-value mean squared error (MSE) against a refitted Gram-PE-form tied control under both robust-only and Muon-aware fitting protocols, while the final Muon-aware recipe improves Basic GMBI further. Paired FP64 polar-factor comparisons confirm gains on matched and held-out GPT-2 Medium inputs, while revealing metric-dependent rankings on unseen spectra. Tests across matrix shapes and restart controls separately probe transfer and finite-precision Gram-state repair. Kernel benchmarks show that compact GMBI graphs can reduce polar-call latency on large rectangular matrices under both compiled and specialized backends. In a nine-method, six-seed GPT-2 Small comparison, the four-layer full-restart Variant achieves the lowest mean validation loss and a slightly lower mean step time than Direct PE, restarted Gram PE, and the two frozen learned-tied realizations; the three-layer Variant has the lowest mean step time.

1 Introduction

Matrix-valued transformations appear inside several large-scale optimizers. Full-matrix Adagrad uses a full-matrix second-moment preconditioner (Duchi et al., 2011). Shampoo extends matrix-root preconditioning to tensor parameters through Kronecker-factored inverse roots (Gupta et al., 2018), while SOAP applies Adam-style updates in the eigenspaces defined by Shampoo statistics (Vyas et al., 2025). K-FAC approximates the Fisher information matrix using Kronecker factors (Martens & Grosse, 2015), and AdaHessian uses stochastic diagonal Hessian estimates (Yao et al., 2021). Muon, documented as an optimizer for hidden-layer matrices (Jordan et al., 2024), applies an approximate polar map to matrix-shaped momentum updates. A related operator-norm steepest-descent perspective is provided in (Bernstein & Newhouse, 2024). This setting changes the numerical objective. An optimizer calls the matrix routine many times per training step, commonly in low precision, so a coarse approximation produced by a small number of dense GPU kernels can be more useful than an asymptotically accurate factorization.

The relevant design requirements are: high parallelism; bounded temporary memory; compatibility with low-precision arithmetic, with this paper focusing on FP16; useful amplification of very small singular values within only a few iterations; and a computation graph that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

compilers can fuse or schedule effectively. QR and SVD remain the reference definitions, but their synchronization and factorization structure make them unattractive as the inner loop of Muon. Polynomial iterations replace those factorizations with matrix multiplications and linear combinations.

1.1 Muon Optimizer and Polar Decomposition

Muon is an alternative to coordinatewise adaptive methods such as Adam and AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). For a matrix parameter W_t, momentum matrix M_t, and stochastic gradient G_t, a simplified update is

$$M_t = \beta M_{t-1} + (1-\beta)G_t, \qquad W_{t+1} = W_t - \eta \,\mathrm{polar}(M_t).$$

If $M = U\Sigma V^\mathsf{T} \in \mathbb{R}^{n\times m}$ with $n \le m$, its row-orthogonal polar factor is the closest row-orthogonal matrix in Frobenius norm (Björck & Bowie, 1971; Higham, 1986):

$$\mathrm{polar}(M) = UV^\mathsf{T} = \operatorname*{arg\,min}_{YY^\mathsf{T}=I_n} \|M - Y\|_\mathrm{F}.$$

For $n > m$, we transpose before applying the algorithm and transpose the output back. Muon and its scalable implementations have been studied for neural-network and language-model training (Jordan et al., 2024; Kimi Team, 2025; Liu et al., 2025). The role of the polar map is to preserve the singular directions of M while mapping each nonzero singular value toward one.

1.2 Polynomial Polar Approximation

For an odd polynomial $p(x) = xh(x^2)$ and $M = U\Sigma V^\mathsf{T}$, exact arithmetic gives $p(M) = Up(\Sigma)V^\mathsf{T}$: the singular vectors are preserved and the singular values are transformed independently. Early inverse-free polynomial iterations for approximate orthogonalization were developed by Kovarik (Kovarik, 1970) and Björck and Bowie (Björck & Bowie, 1971). Their familiar cubic specialization, commonly referred to as the Newton–Schulz polar iteration, maps x by $p_\mathrm{NS}(x) = \frac{3}{2}x - \frac{1}{2}x^3$; see Higham (Higham, 2008) for a modern treatment. Muon commonly uses an aggressive repeated quintic (Jordan et al., 2024); layer-dependent quintics and Chebyshev constructions improve fixed-depth shaping (Cesista et al., 2025; Grishina et al., 2025); and Polar Express (PE) optimizes a minimax sequence over a prescribed interval (Amsel et al., 2026). Gram Newton–Schulz evaluates such odd-polynomial sequences through a square state and accumulator (Zhang et al., 2026). This square-state realization motivates GMBI's reuse of S, S^2, and intermediate factors. Appendix A gives the baseline equations and the fixed Polar Express coefficients used here.

1.3 Our Contributions

Our central contribution is a decoupled polynomial state-space parameterization. We write each layer as a state transition $S_i = g_i(S_{i-1})$ and output update $Q_i = f_i(S_{i-1})Q_{i-1}$. A conventional Gram quintic ties them by $g_i(s) = sf_i(s)^2$; GMBI learns them separately while sharing the same square-state intermediates. We formulate basic and variant recurrences, define an FP16 full restart that rebuilds a true Gram state, compare robust and Muon-aware coefficient selection against a learned tied control, and evaluate approximation, isolated latency, and six-seed GPT-2 Small training.

Decoupling is the architectural contribution: it creates additional polynomial flexibility under the stated graph and reuse constraints. In exact arithmetic the final GMBI map remains an odd polynomial map $X \mapsto q(XX^\mathsf{T})X$; the novelty lies in the constrained state-space parameterization rather than an unrestricted polynomial function class. The short graphs test whether decoupled state transition and readout can compensate for depth. We do not infer cost from four layers versus five layers. Instead, we give separate dense-GEMM and symmetry-aware operation counts as functions of matrix aspect ratio. The conditional 20% figure for square inputs applies only when a specialized kernel computes one triangle of every theoretically symmetric square output and mirrors it; ordinary dense matrix multiplication receives no such saving. The five-layer restart graphs instead target robustness and approximation accuracy; they do not claim a lower layer count than Polar Express.

Mathematical notation. $X \in \mathbb{R}^{n\times m}$ denotes the normalized rectangular input, with $n \le m$ after an optional transpose. $U\Sigma V^{\mathsf{T}}$ is its singular value decomposition, $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, and I_n is the identity. $R = XX^{\mathsf{T}}$ denotes a true Gram matrix. S_i denotes a learned auxiliary square state, which need not be a Gram matrix or be positive semidefinite. Q_i is the square accumulator and the rectangular output is $Q_T X$. We write $B_i = S_i^2$ and

$$F_{i,j} = a_{i,j} I_n + b_{i,j} S_{i-1} + c_{i,j} B_{i-1}.$$

The variant additionally uses d_i and e_i. T is the layer count, and a restart before layer 3 is abbreviated in prose as a full layer-3 restart. Matrix inputs are normalized as $X = M/(1.02\|M\|_{\mathrm{F}} + 10^{-6})$; Section 4 states the evaluated arithmetic precisions, while spectral diagnostics use FP32 unless a recorded SVD fallback is required.

2 Gram Newton-Schulz and Optimal Odd Polynomial Approximation

Let $X_0 = M/(1.02\|M\|_{\mathrm{F}} + 10^{-6})$. The factor 1.02 provides a normalization margin; 10^{-6} keeps the denominator positive, including at zero input. For an odd polynomial $p_t(x) = xh_t(x^2)$, a Gram realization begins with

$$R_0 = X_0 X_0^{\mathsf{T}}, \qquad Q_0 = I_n,$$

and updates

$$Z_t = h_t(R_{t-1}), \quad R_t = Z_t R_{t-1} Z_t, \quad Q_t = Z_t Q_{t-1}, \quad X_T = Q_T X_0.$$

In exact arithmetic all terms are polynomials of R_0 and commute. In floating point, multiplication order matters and the computed state can lose symmetry or positive semidefiniteness. Therefore the word "Gram" describes the construction and the initial state; it does not make every learned intermediate a true Gram matrix.

The same recurrence exposes the coupling that motivates GMBI. On a scalar Gram eigenvalue s, let $f_t(s) = h_t(s)$. The accumulator is multiplied by $f_t(s)$, while Gram consistency forces the next state map to be

$$g_t(s) = s f_t(s)^2.$$

Thus choosing the direct quintic f_t also fixes the state transition g_t. Retraining the coefficients of f_t does not remove this constraint.

Polar Express chooses each h_t to control a worst-case scalar error over a known spectral interval (Amsel et al., 2026). Such a minimax objective supplies a useful guard against blow-up anywhere in the interval, but it does not model the nonuniform spectra encountered by Muon. Learning the coefficients of each direct quintic would adapt the same tied update to a sampled distribution; coefficient learning alone is therefore not the distinction claimed here. GMBI changes the state-space parameterization by learning the state-transition and output maps separately, subject to shared intermediates and an explicit matrix-operation budget. Its fitting objective combines a minimax-style prefix objective with empirical spectrum weighting and explicit guard points. This preserves interval coverage while assigning additional weight to the small-singular-value tail.

Cost interpretation. Let $\rho = m/n \ge 1$ and measure work in full $n \times n$ GEMM equivalents. Five-layer Direct PE and four-layer Basic GMBI cost, respectively,

$$(C_{\mathrm{PE}}^{\mathrm{dense}}, C_{\mathrm{GMBI4}}^{\mathrm{dense}}) = (10\rho + 5, 2\rho + 13), \qquad (C_{\mathrm{PE}}^{\mathrm{sym}}, C_{\mathrm{GMBI4}}^{\mathrm{sym}}) = (7.5\rho + 2.5, 1.5\rho + 6.5).$$

Thus dense square execution gives no arithmetic saving (both cost 15 at $\rho = 1$); the often quoted 20% reduction requires a triangular-output kernel for every theoretically symmetric product. Rectangular inputs can favor GMBI because repeated rectangular products are replaced by smaller square-state work. Appendix B gives the complete accounting and finite-precision qualifications.

3 GMBI Algorithm Framework

Let $S_{i-1} \in \mathbb{R}^{n\times n}$ be the auxiliary square state and define

$$F_{i,j} = a_{i,j}I + b_{i,j}S_{i-1} + c_{i,j}S_{i-1}^2, \qquad j = 1,\dots,4.$$

The basic layer forms $H_i = F_{i,1}F_{i,2}$ and updates

$$S_i = H_iF_{i,3}, \qquad Q_i = F_{i,4}Q_{i-1}.$$

The final S_i need not be computed. The output is Q_TX_0. The square state is a compact carrier of a learned polynomial in the initial Gram matrix; except after initialization or restart, it is not asserted to equal the Gram matrix of the current rectangular iterate.

The GMBI variant enriches the decoupled state-space update without introducing a historical-state method. It uses the scalar coefficients d_i and e_i in the update

$$S_i = H_i(F_{i,3} + d_iH_i), \qquad Q_i = (F_{i,4} + e_iH_i)Q_{i-1}.$$

The d_i path modifies only the state transition g_i, whereas the e_i path modifies only the readout f_i, making the two reuse directions independently controllable. We study a three-layer form for low nominal depth and a five-layer gain-limited form for restart experiments. We use the plain name "GMBI variant" throughout because no second variant is retained.

3.1 Decoupled polynomial state-space parameterization

For scalar notation define

$$\phi_{i,j}(s) = a_{i,j} + b_{i,j}s + c_{i,j}s^2, \qquad h_i(s) = \phi_{i,1}(s)\phi_{i,2}(s).$$

The basic GMBI layer realizes

$$g_i(s) = h_i(s)\phi_{i,3}(s), \qquad f_i(s) = \phi_{i,4}(s),$$

whereas the variant realizes

$$g_i(s) = h_i(s)\big(\phi_{i,3}(s) + d_ih_i(s)\big), \qquad f_i(s) = \phi_{i,4}(s) + e_ih_i(s).$$

Consequently, $S_i = g_i(S_{i-1})$ and $Q_i = f_i(S_{i-1})Q_{i-1}$ are not constrained by $g_i(s) = sf_i(s)^2$. This is the defining decoupling. It lets fitting modify the auxiliary-state trajectory without imposing the same change on the output factor, or modify the output factor without forcing a Gram-consistent state update. The two maps remain computationally coupled through their shared use of S, S^2, and $H = F_1F_2$; GMBI therefore decouples their coefficients without paying for two independent polynomial bases.

3.2 Why use these two GMBI forms?

We restrict attention to graphs that form S^2 once per layer, reuse S, S^2, and H, keep intermediate states square, and defer square-by-rectangular multiplication until output or restart. Basic GMBI assigns separate factors to state transport and readout; the Variant adds $d_iH_i^2$ and e_iH_i reuse paths without another polynomial basis. These are preferred designs within this budget, not globally optimal matrix polynomials. Complete pseudocode and implementation details appear in Appendix F.

3.3 Full restart before layer 3

FP16 errors can make the auxiliary state inconsistent with the rectangular iterate. A full restart preserves the prefix action while rebuilding a physical Gram state:

$$X \leftarrow Q_2X_0, \qquad S \leftarrow XX^{\mathsf{T}}, \qquad Q \leftarrow I_n,$$

Here X, S, Q are updated in place, as in Algorithm 1; layers 3–5 then act on the restarted X. Between restarts, S_i follows the learned g_i and need not equal the Gram matrix implied by Q_iX; the restart restores this consistency. We apply no additional normalization at restart.

Algorithm 1: Five-layer GMBI with full restart before layer 3; unused final-state factors are pruned.

Input: M, five coefficient layers; basic or Variant update

$X \leftarrow M/(1.02\|M\|_{\mathrm{F}} + 10^{-6})$; $S \leftarrow XX^{\mathsf{T}}$; $Q \leftarrow I$;
for $i = 1, \ldots, 5$ do
  if $i = 3$ then
    $X \leftarrow QX$; $S \leftarrow XX^{\mathsf{T}}$; $Q \leftarrow I$;
  end
  $F_j \leftarrow a_{i,j} I + b_{i,j} S + c_{i,j} S^2$ for required j;
  $H \leftarrow F_1 F_2$ when required;
  $Q \leftarrow F_4 Q$ (basic) or $(F_4 + e_i H)Q$ (Variant);
  if $i < 5$ and $i \neq 2$ then
    $S \leftarrow HF_3$ (basic) or $H(F_3 + d_i H)$ (Variant);
  end
end
return QX;

3.4 Coefficient fitting

We fit and freeze layers 1–2 before fitting layers 3–5. Robust-only fitting uses the prefix-induced scalar interval with dense guards; Muon-aware fitting executes the prefix in FP16, rebuilds true restart Gram states, and fits the suffix on their realized spectra. Muon-aware fitting changes both the empirical support and the post-restart objective; it is a complete fitting protocol rather than an isolated data-distribution ablation. Every GPT-2 Medium matrix remains test-only. The tied learned control enforces $g_i(s) = sf_i(s)^2$; published Direct/Gram PE coefficients remain fixed external controls. Appendix G specifies the final fitting schedules, objectives, and selection rules.

4 Numerical Experiments

The experiments answer four questions in order. Section 4.1 tests state–output decoupling and Muon-aware fitting under the proposed phasewise protocol and a jointly fitted five-layer control. Section 4.2 compares nine methods, including direct and Gram realizations of a frozen learned tied polynomial, on held-out GPT-2 Medium updates and synthetic spectral families using the same compiled PyTorch realization. Section 4.3 compares torch.compile and Dao-AILab QuACK for the same nine methods across six matrix shapes on H100. Section 4.4 reports downstream GPT-2 training. Approximation error and runtime are not mixed in the same table.

4.1 Five-layer comparison: decoupling, calibration, and fitting protocols

This section asks whether relaxing the constraint between state propagation and output accumulation improves polar approximation at fixed depth, restart position, and principal matrix-multiplication budget, and how coefficient fitting affects that improvement. To evaluate the basic structure directly, we compare Basic GMBI with a refitted Gram-PE-form tied control, without the Variant's additional reuse paths.

The tied control uses a quadratic $f_i(s)$ and enforces

$$g_i(s) = sf_i(s)^2.$$

Its recurrence has the Gram Polar Express form, but its coefficients are optimized under this section's fitting protocols rather than copied from the published PE sequence. Basic GMBI separately parameterizes the state and output maps. In fact, imposing

$$\phi_{i,1}(s) = s, \qquad \phi_{i,2}(s) = \phi_{i,3}(s) = \phi_{i,4}(s) = f_i(s)$$

on Basic recovers the tied recurrence in exact arithmetic. Relaxing this Gram-consistency constraint allows the state trajectory to adjust independently of the output factor. Decoupling necessarily changes the admissible coefficient space. We therefore evaluate the tied and

decoupled recurrences as complete graph parameterizations under matched depth, restart placement, and matrix-operation budget.

All configurations use five layers and full restart before layer 3. The experiment crosses two parameterizations, two calibration settings, and two complete fitting protocols. Robust-only uses spectral support and interval guards without GPT-2 updates; Muon-aware also calibrates on GPT-2 Small updates and uses an empirical suffix objective with interval protection. Phasewise fitting freezes the first two layers before optimizing the last three: the robust-only suffix uses the prefix-induced scalar interval, whereas the Muon-aware suffix uses spectra from FP16-materialized prefix outputs. The joint control optimizes all five layers through scalar spectral rollouts. Phasewise and joint fitting also differ in objective, numerical execution, calibration-state construction, and optimization settings; their comparison evaluates the complete recipes. Coefficient selection and calibration construction are described in Appendix G; the controlled comparison fixes the matrix-execution budget.

Evaluation uses 27 matched-family matrices, 92 unseen-spectrum conditions, and 28 held-out GPT-2 Medium updates. All methods receive identical normalized, FP16-rounded inputs, and Medium matrices never enter fitting or selection. For $Y \in \mathbb{R}^{n\times m}$, $n \le m$, we measure

$$E_{\mathrm{sv}}(Y) = \frac{1}{n}\sum_{j=1}^{n}(\sigma_j(Y) - 1)^2,$$

then average this singular-value mean squared error (MSE) equally over matrices in each test set. This metric measures how closely the output singular values approach one. It does not by itself test the polar factor's direction: for an orthogonal $R \neq I_n$, $Y = R\,\mathrm{polar}(X)$ can have $E_{\mathrm{sv}}(Y) = 0$ and $YY^{\mathsf{T}} = I_n$ while differing from $\mathrm{polar}(X)$. We therefore pair the existing spectral results with FP64-reference output error, output angle, orthogonality residual, and upper-tail diagnostics in Appendix J.4.

Table 1: Basic GMBI versus the learned tied control with five layers and full R3. Joint and phasewise denote the two complete fitting protocols. Values are mean singular-value MSE on identical FP16 inputs; lower is better. Bold marks the minimum in each test set. Final fitting settings are given in Appendix G.

test set	fitting	state/output	robust-only	Muon-aware
matched family (27)	joint control	tied	0.335330	0.327960
	joint control	Basic GMBI	0.326983	0.314352
	phasewise (ours)	tied	0.326806	0.331365
	phasewise (ours)	Basic GMBI	0.323482	**0.279813**
unseen spectra (92)	joint control	tied	0.085612	0.085200
	joint control	Basic GMBI	0.082294	**0.080805**
	phasewise (ours)	tied	0.084003	0.085294
	phasewise (ours)	Basic GMBI	0.082741	0.081073
held-out Medium (28)	joint control	tied	0.146959	0.133066
	joint control	Basic GMBI	0.133926	0.126544
	phasewise (ours)	tied	0.131538	0.134899
	phasewise (ours)	Basic GMBI	0.129474	**0.112767**

Fitting for application-relevant spectra. Phasewise Muon-aware Basic GMBI achieves the lowest mean error on matched families and held-out Medium updates. Relative to the corresponding joint fit, its MSE is lower by 11.0% and 10.9%, respectively. Its unseen-spectrum mean differs from the joint fit by only 0.33% (0.081073 versus 0.080805), while improving on phasewise robust-only by 2.0%. Thus the final phasewise recipe combines larger gains on application-relevant distributions with competitive average accuracy on broader spectral conditions. Muon-aware fitting improves Basic under both fitting protocols in all three sets; for phasewise fitting, the reductions relative to robust-only are 13.5%, 2.0%, and 12.9%.

Benefits of the decoupled parameterization. Across all twelve corresponding comparisons of calibration setting, fitting protocol, and test set, Basic GMBI has lower mean error than

the tied control. Under phasewise robust-only fitting, the reductions are 1.0%, 1.5%, and 1.6%; under the final phasewise Muon-aware recipe, they are 15.6%, 4.9%, and 16.4%. These results support the practical value of separately adjustable state and output maps within the evaluated matrix-operation budget. The larger gains on matched families and held-out Medium updates show that the benefit extends to a model scale not used for calibration.

The frozen Muon-aware phasewise tied coefficients give direct-graph MSE 0.331429/0.085078/0.134911 on matched/unseen/Medium inputs, nearly matching their Gram realization (0.331365/0.085294/0.134899). Basic remains better on matched and Medium inputs. A separate FP64 direct–Gram parity check differs by at most 6.01×10^{-15}.

The paired polar-factor audit confirms the final Basic recipe's advantage over its tied counterpart on matched families (0.4633 versus 0.5035) and held-out Medium updates (0.2530 versus 0.2760) in normalized Frobenius error. On unseen spectra, however, Basic robust joint has the lowest mean polar-factor error (0.1998); the final Basic recipe gives 0.2136 despite its lower singular-value MSE than Basic robust phasewise. The final recipe's unseen-set 95th-percentile polar-factor error is 0.5475 versus 0.5822 for robust phasewise. Thus the broad unseen-spectrum ranking depends on the chosen error metric; Appendix J.4 reports all paired methods and tails.

Restart is controlled separately with the same published Polar Express coefficients. On its separate synthetic control set, Direct PE, Gram PE without restart, and Gram PE with full R3 give FP16 MSE 0.08238, 0.08955, and 0.08259. On the Medium matrices they give 0.13661, 0.79756, and 0.13652; two no-restart outputs are non-finite. An independent FP64 gate gives maximum relative disagreement 4.50×10^{-14} between the three realizations. Restart therefore repairs finite-precision Gram-state error. Since both learned controls in Table 1 use full R3, the Basic–tied difference is not a comparison of restart versus no restart.

An independent FP64/FP32/FP16/BF16 matrix screen in Appendix E delineates the precision dependence of ten frozen methods. It is a numerical stress test, not a second tied-versus-decoupled fitting ablation.

4.2 Actual-matrix approximation of nine methods

We compare nine frozen algorithms under identical FP16 normalization and separately captured torch.compile graphs. The real groups contain four nonzero Small and eight held-out Medium updates; the synthetic screen uses two held-out seeds at 384×1536. Learned direct and Gram learned direct use the same frozen tied coefficients but realize the polynomial through rectangular iterates and Gram states, respectively. Table 2 reports mean per-matrix singular-value MSE; Appendix J specifies capture stages and spectral families.

Table 2: FP16 singular-value MSE on actual and synthetic matrices using compiled PyTorch. Small and Medium contain 4 and 8 nonzero captured Muon updates. Each synthetic family uses two held-out seeds at shape 384×1536. Lower is better.

method	GPT-2 Small	GPT-2 Medium	Gaussian	tail/low rank	unseen spectra
Direct Polar Express, 5	0.165181	0.232886	0.008670	0.639260	0.071551
Gram Polar Express, 5, full R3	0.165221	0.232914	0.008663	0.638331	0.071604
Learned direct, 5	0.162908	0.230581	0.006473	0.638025	0.069196
Gram learned direct, 5, full R3	0.162941	0.230595	0.006476	0.637172	0.069181
GMBI Basic, 4, no restart	0.385479	0.457184	0.000704	0.711260	0.118906
GMBI Variant, 3, no restart	0.449559	0.519780	0.006676	0.727647	0.148425
GMBI Variant, 4, full R3	0.196531	0.268131	0.001642	0.652987	0.072073
GMBI Basic, 5, full R3	0.135631	0.193273	0.003501	0.611978	0.067114
GMBI Variant, 5, full R3	0.085409	0.132806	0.000301	0.572999	0.050103

The five-layer restart Variant has the lowest MSE in all five groups; five-layer Basic also beats both published PE controls throughout. The two learned tied realizations improve fixed Direct PE in all five groups but trail both five-layer GMBI graphs. Their differences are small, as expected from their shared exact polynomial; the remaining gap measures finite-precision realization effects. Variant depth offers matrix accuracy at five layers, low polar-call latency at three, and an intermediate four-layer restart choice with the best observed

Small-training loss. On captured updates, four-layer restart improves on shorter GMBI graphs without uniformly beating either PE control. Direct and restarted Gram PE remain nearly identical, consistent with their shared exact map.

The FP64-reference replay also finds the five-layer restart Variant lowest in mean normalized polar-factor error in all five groups. On held-out Medium updates it scores 0.2957 versus 0.4254 for Direct PE; across all 26 nonzero inputs its 95th percentile is 0.8437 versus 0.9028. Its worst polar-factor error is still 0.9030 on a near-low-rank input, so the average advantage does not imply uniformly accurate factors. Appendix J.4 gives the paired output angles, orthogonality residuals, and full tail summary.

The ranking also survives a frozen six-configuration shape screen: the five-layer restart Variant gives the lowest mean MSE on held-out Medium, square, and rectangular groups (0.132811, 0.004204, and 0.000433).

4.3 Speed comparison across matrix shapes and backends

We time one complete FP16 polar call on an H100 80GB with PyTorch 2.7.1/CUDA 12.8. Within each shape, all nine methods receive the same deterministic input under both backends. Table 3 reports CUDA-event medians from 154 synchronized observations per method, backend, and shape; all outputs are finite and every timed graph passes an explicit CUDA-Graph replay probe. The generation and capture protocol is in Appendix J.

Table 3: Nine-method H100 FP16 polar-call median latency (ms), ordered by total input elements. Each cell reports compiled PyTorch / Dao-AILab QuACK; bold marks the fastest within each column and backend. Basic and Variant denote GMBI; R3 denotes full restart. Shapes are batch$\times m \times n$; lower is better.

method	4×4096^2	$2\times4096\times8192$	$16\times1024\times4096$	2×4096^2	32×1024^2	$11\times768\times3072$
Direct PE (5)	14.208/9.604	11.576/8.305	3.547/2.458	6.929/4.928	2.533/1.841	1.198/0.850
Gram PE (5,R3)	16.014/10.221	9.710/6.731	2.036/1.863	7.905/5.259	2.441/2.087	0.675/0.619
Learned direct (5)	14.142/9.527	11.682/7.974	3.553/2.292	6.964/4.741	2.573/1.757	1.201/0.854
Gram learned direct (5,R3)	16.097/10.304	9.589/6.776	2.048/1.889	7.766/5.430	2.449/2.092	0.680/0.622
Basic (4)	14.853/9.295	8.226/5.701	1.886/1.706	7.315/4.951	2.678/2.298	0.652/0.649
Variant (3)	11.145/7.178	6.455/4.466	1.557/1.396	5.549/3.758	2.056/1.769	0.550/0.500
Variant (4,R3)	14.934/9.845	9.303/6.405	2.143/1.916	7.365/5.062	2.648/2.371	0.729/0.703
Basic (5,R3)	18.283/12.002	10.898/7.592	2.464/2.234	9.042/6.186	3.277/2.840	0.834/0.758
Variant (5,R3)	18.905/12.174	10.967/7.749	2.484/2.257	9.183/6.420	3.292/2.842	0.834/0.758

The three-layer Variant is fastest in all six columns under compiled PyTorch and five of six under Dao; on 32×1024^2, learned direct leads Dao by 0.012 ms. The four-layer restart Variant beats restarted Gram PE on every selected 4096-state shape under both backends: on 4×4096^2, $2 \times 4096 \times 8192$, and 2×4096^2, respectively, its compiled latency is lower by 6.74%, 4.19%, and 6.83%, and its Dao latency by 3.68%, 4.84%, and 3.75%. It does not beat Gram PE on the two 1024-state shapes or on $11 \times 768 \times 3072$, so the advantage is tied to sufficiently large Gram states rather than matrix size alone. These are isolated polar calls, not frequency-weighted GPT-2 training throughput.

All nine methods use the same frozen coefficients, per-shape seeds, six excluded warm-ups, and 14 blocks of 11 synchronized CUDA-event samples. Frozen Python float coefficients are specialized at graph capture, and a profiler gate requires three CUDA-Graph launches for three probe calls before timing. Dao GMBI materializes a two-stage symmetric S^2; PE retains its corresponding kernels. Backend comparisons therefore measure whole implementations.

4.4 Original-Muon GPT-2 Small training

We use the original per-parameter Muon path with a 12-layer, width-768 GPT-2 Small model, context 1024, FineWeb (Penedo et al., 2024), two H100s, mixed precision, and 5100 steps. Six fixed seeds (842641, 193171, 284759, 376127, 468593, 759431) share batches across methods; method order is rotated. Table 4 reports the 54 compiled-PyTorch runs at

the common step-5100 endpoint. train s excludes the first ten warm-up steps and validation/checkpoint intervals; step ms= 1000 train s/5090, rounded to two decimals in each run log before aggregation. The matched Dao backend results are reported in Appendix I.

Table 4: GPT-2 Small results under the original per-parameter Muon implementation and compiled PyTorch backend. Values are mean ± sample standard deviation over six matched seeds. Every row reaches step 5100. Lower is better; bold marks the lowest mean validation loss.

method	validation loss at 5100	train s	step ms
Direct Polar Express, 5	3.280700 ± 0.000865	2806.995 ± 8.413	551.475 ± 1.653
Gram Polar Express, 5, full R3	3.280550 ± 0.000894	2803.458 ± 9.270	550.777 ± 1.820
Learned direct, 5	3.281033 ± 0.001275	2807.780 ± 18.167	551.627 ± 3.569
Gram learned direct, 5, full R3	3.280833 ± 0.001347	2804.386 ± 18.559	550.960 ± 3.646
GMBI Basic, 4, no restart	3.280900 ± 0.002260	2805.360 ± 8.945	551.150 ± 1.758
GMBI Variant, 3, no restart	3.282567 ± 0.001853	2799.490 ± 12.164	550.000 ± 2.390
GMBI Variant, 4, full R3	$\mathbf{3.279417 \pm 0.000527}$	2802.733 ± 11.418	550.635 ± 2.245
GMBI Basic, 5, full R3	3.282583 ± 0.001011	2803.334 ± 11.614	550.753 ± 2.283
GMBI Variant, 5, full R3	3.281233 ± 0.001477	2805.897 ± 6.787	551.257 ± 1.336

All 54 compiled runs reach step 5100 with exit code zero and finite validation loss. The four-layer GMBI Variant with full R3 achieves the lowest mean validation loss of the nine methods, 3.279417: it improves on Direct PE (3.280700) in all six matched seeds and on restarted Gram PE (3.280550) in five of six. Under compiled PyTorch, the four-layer restart Variant outperforms Direct PE at all eight late-training validation checkpoints (steps 4750–5100, every 50 steps) in each of the six matched seeds. The final paired loss difference against Direct PE is -0.001283 (nominal 95% CI $[-0.002255, -0.000311]$); averaging the eight checkpoints within each seed gives -0.001329 ($[-0.002241, -0.000417]$). This small but persistent validation-loss improvement supports a promising downstream accuracy benefit. Appendix K reports individual seeds and late-training curves. Its mean step time, 550.635 ms, is 0.840 ms below Direct PE and 0.142 ms below Gram PE. The three-layer no-restart Variant is fastest overall at 550.000 ms per step, with mean loss 3.282567. The nominal paired timing intervals span zero for both compiled-PE comparisons (Appendix K). The small numerical timing differences are consistent with limited Amdahl headroom: polar evaluation occupies only part of a training step, while forward and backward computation, communication, and the remaining optimizer work are unchanged.

The frozen learned-direct controls closely track their fixed PE counterparts: their compiled mean losses are 3.281033 and 3.280833, and neither improves on the four-layer restart Variant. The paired backend comparison is reported in Appendix I. Dao is 2.502–4.497 ms per step slower than compiled PyTorch for the Gram-state methods, while Direct PE is 2.485 ms per step slower. This per-parameter training path processes predominantly small, heterogeneous matrices through sequential calls, which amortize kernel-launch, dispatch, and composition overhead less effectively than the large batched configurations in Section 4.3.

5 Conclusion and Future Work

GMBI decouples Gram-state propagation and output accumulation yet remains an odd polynomial. Basic lowers FP16 MSE against a refitted tied control (Table 1) and matched/Medium polar-factor error. Restart repairs state drift; unseen rankings depend on the metric.

Both frozen tied controls improve fixed Direct PE across five matrix MSE groups; both five-layer GMBI graphs improve further. Variant T5 leads nine methods in mean singular-value error and the audited methods in mean polar-factor error. T3 is fastest on six compiled shapes, five Dao shapes, and in Small training. T4 R3 has the lowest six-seed Small loss, improves every late validation point over Direct PE, and beats Gram PE on three tested 4096-state calls under both backends. BF16 stability, larger models, and joint graph/kernel design remain open.

AI Use Statement

Generative AI tools assisted with hypothesis refinement, experimental design, software implementation, result checking and interpretation, and manuscript drafting and editing. The literature search was conducted by the authors, and the core innovations were conceived by the authors. All AI-assisted code, calculations, experimental records, and claims were reviewed and validated by the authors, who take responsibility for the final content.

References

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=yRtgZ1K8hO.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. In OPT 2024: Workshop on Optimization for Machine Learning, 2024. URL https://arxiv.org/abs/2409.20325.

Åke Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. SIAM Journal on Numerical Analysis, 8(2):358–364, 1971. doi: 10.1137/0708036.

Franz Louis Cesista, Jiacheng You, and Keller Jordan. Squeezing 1–2% efficiency gains out of muon by optimizing the Newton–Schulz coefficients, February 2025. URL https://leloykun.github.io/ponder/muon-opt-coeffs/. Technical blog post.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(61):2121–2159, 2011. URL https://jmlr.org/papers/v12/duchi11a.html.

Ekaterina Grishina, Matvey Smirnov, and Maxim Rakhuba. Accelerating Newton–Schulz iteration for orthogonalization via Chebyshev-type polynomials. arXiv preprint arXiv:2506.10935, 2025. doi: 10.48550/arXiv.2506.10935. URL https://arxiv.org/abs/2506.10935.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 1842–1850. PMLR, 2018. URL https://proceedings.mlr.press/v80/gupta18a.html.

Nicholas J. Higham. Computing the polar decomposition—with applications. SIAM Journal on Scientific and Statistical Computing, 7(4):1160–1174, 1986. doi: 10.1137/0907079.

Nicholas J. Higham. Functions of Matrices: Theory and Computation. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2008. doi: 10.1137/1.9780898717778.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. Technical blog post.

Kimi Team. Kimi K2: Open agentic intelligence. arXiv preprint arXiv:2507.20534, 2025. doi: 10.48550/arXiv.2507.20534. URL https://arxiv.org/abs/2507.20534.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations, 2015. URL https://arxiv.org/abs/1412.6980.

Zdislav Kovarik. Some iterative methods for improving orthonormality. SIAM Journal on Numerical Analysis, 7(3):386–389, 1970. doi: 10.1137/0707031.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. arXiv preprint arXiv:2502.16982, 2025. doi: 10.48550/arXiv.2502.16982. URL https://arxiv.org/abs/2502.16982.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pp. 2408–2417. PMLR, 2015. URL https://proceedings.mlr.press/v37/martens15.html.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In Advances in Neural Information Processing Systems, volume 37, 2024. doi: 10.52202/079017-0970. URL https://papers.nips.cc/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael W. Mahoney. ADAHESSIAN: An adaptive second order optimizer for machine learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(12):10665–10673, 2021. doi: 10.1609/aaai.v35i12.17275. URL https://ojs.aaai.org/index.php/AAAI/article/view/17275.

Jack Zhang, Noah Amsel, Berlin Chen, and Tri Dao. Gram Newton–Schulz: A fast, hardware-aware Newton–Schulz algorithm for muon, 2026. URL https://dao-lab.ai/blog/2026/gram-newton-schulz/. Dao AI Lab technical blog post.

A Polynomial Baselines

Classical polar Newton–Schulz uses $p_{\mathrm{NS}}(x) = \frac{3}{2}x - \frac{1}{2}x^3$ (Kovarik, 1970; Björck & Bowie, 1971; Higham, 2008). Muon's repeated Jordan quintic is $p_J(x) = 3.4445x - 4.7750x^3 + 2.0315x^5$ (Jordan et al., 2024). Layer-dependent quintics write $p_t(x) = a_t x + b_t x^3 + c_t x^5$ (Cesista et al., 2025); Polar Express chooses such maps by a minimax criterion (Amsel et al., 2026). The fixed five-layer Polar Express sequence used in our external controls is

t	a_t	b_t	c_t
1	8.156555	−22.483293	15.878770
2	4.042930	−2.808917	0.500018
3	3.891668	−2.772484	0.506065
4	3.285754	−2.368129	0.464490
5	2.346541	−1.709782	0.423236

with $X_{t+1} = a_t X_t + (b_t R_t + c_t R_t^2) X_t$ and $R_t = X_t X_t^{\mathsf{T}}$. CANS distinguishes convergent and aggressive inexact Chebyshev-type maps (Grishina et al., 2025); the latter are relevant when only a few layers are available.

B Detailed Cost Accounting

Use the $2n^3$ FLOPs of a full $n \times n$ GEMM as one unit and let $\rho = m/n \geq 1$. A Direct PE layer forms a rectangular Gram product, one square product, and a square-by-rectangular update, so $C_{\mathrm{PE}}^{\mathrm{dense}} = 5(\rho + 1 + \rho) = 10\rho + 5$. After pruning the unused final state and identity-accumulator products, four-layer Basic GMBI uses the initial Gram product, 13 square products, and one final rectangular product, so $C_{\mathrm{GMBI4}}^{\mathrm{dense}} = 2\rho + 13$.

If every exact-arithmetic symmetric output is computed by a triangular-output kernel and mirrored, Gram and square products cost half their dense equivalents while square-by-rectangular products do not. Hence $C_{\mathrm{PE}}^{\mathrm{sym}} = 7.5\rho + 2.5$ and $C_{\mathrm{GMBI4}}^{\mathrm{sym}} = 1.5\rho + 6.5$. At $\rho = 1$ the dense costs are both 15, whereas the symmetry-aware costs are 10 and 8. The latter reduction does not follow from symmetric inputs alone: ordinary PyTorch/cuBLAS GEMM still computes the full output. Moreover, finite-precision factors need not commute, so mirroring one triangle defines a numerical implementation that must be validated together with latency. Launch overhead, linear combinations, memory traffic, restart products, and the rest of training are outside these ratios.

C Gram Newton–Schulz Implementation Details

The recurrence in Section 2 can be implemented without materializing an identity matrix. Diagonal constants are inserted during the output write of a symmetric linear-combination kernel. We symmetrize only when a diagnostic eigendecomposition requires it; repeatedly replacing a computed matrix by $(A + A^{\mathsf{T}})/2$ changes the implemented polynomial and adds memory traffic. Timings in the main text include the actual backend selected for the run.

For completeness, Algorithm 2 states the ordinary Gram realization of a sequence of odd polynomials. It is algebraically equivalent to applying the composite odd polynomial to X_0 in exact arithmetic. This equivalence explains why a square-state method can return a rectangular polar approximation after only one final square-by-rectangular product.

Algorithm 2: General Gram realization of odd polynomial iteration.

Input: $X_0 \in \mathbb{R}^{n \times m}$ and $h_1, \ldots, h_T$
Output: X_T

$R \leftarrow X_0 X_0^{\mathsf{T}}$; $Q \leftarrow I_n$;
for $t = 1, \ldots, T$ do
  $Z \leftarrow h_t(R)$;
  $Q \leftarrow ZQ$;
  if $t < T$ then
    $R \leftarrow ZRZ$;
  end
end
return QX_0;

The basic GMBI state update requires one product for S^2, one for F_1F_2, and one for HF_3 when a next state is needed. Updating a nontrivial accumulator requires another square product; the first accumulator factor and the final unused state permit graph-specific savings. The variant reuses H but adds its d_iH and e_iH paths through linear combinations. A restart additionally materializes QX and recomputes XX^{T}. These details are why layer count alone cannot predict runtime, even though it remains a useful theoretical depth measure.

Implicit diagonal corrections. Let $\widetilde{F}_j = b_jS + c_jS^2$. Rather than first writing four dense matrices $F_j = \widetilde{F}_j + a_jI$, a kernel may load each off-diagonal element once and apply the diagonal constant only when row and column indices coincide. For the first product,

$$H = \widetilde{F}_1\widetilde{F}_2 + a_2\widetilde{F}_1 + a_1\widetilde{F}_2 + a_1a_2I.$$

One may also compute the explicit factors and call a symmetric linear-combination kernel afterward. Which form is faster depends on matrix size, compiler fusion, register pressure,

and kernel launch overhead; the paper therefore treats both as implementation alternatives rather than algebraic speed guarantees.

D FP16 Stability Diagnostics

For each serious run we scan logs and output tensors for NaN, Inf, traceback, or an explicit numerical-explosion marker. Matrix tests additionally record the smallest and largest output singular values and the normalized row-orthogonality error $\|YY^{\mathsf{T}} - I\|_{\mathrm{F}}/\sqrt{n}$. The 90-run matrix suite and the formal GPT-2 queues reported in this paper passed these finite-value gates. This establishes empirical stability only for the tested shapes, spectra, seeds, and hardware; it is not a proof for arbitrary inputs.

Writing the computed state as $\widehat{S}_i = S_i + E_i$, a first-order polynomial perturbation satisfies

$$F(\widehat{S}_i) - F(S_i) \approx bE_i + c(S_iE_i + E_iS_i).$$

Consequently, $\|E_i\|$ is amplified by approximately $|b| + 2|c|\|S_i\|$ before the factor products are considered. Products of several such factors can magnify both the symmetric error and the skew component. The accumulator then converts the square-state error into a direction error in the rectangular output.

Table 5: Observed low-precision failure mechanisms and the corresponding diagnostic.

mechanism	propagation	diagnostic
rounded initial Gram	small negative eigenvalues of the FP16 symmetric part	compare FP16 and FP32 XX^{T} spectra
intrinsic learned state	an auxiliary polynomial may legitimately become indefinite	compare against the same graph in high precision
eigenvector drift	noncommuting roundoff changes later factor directions	record skew norm and output-angle error
coefficient gain	a large polynomial derivative magnifies state error	sweep guard points and actual matrices
restart overflow	a large prefix output enters the suffix Gram	record prefix singular extrema before fitting

These cases must not be conflated. In particular, projecting every auxiliary state onto the positive-semidefinite cone changes the learned polynomial and was not adopted. Full restart is used because it restores a physical Gram state while preserving the prefix transformation.

E Cross-Precision Accuracy and Stability

We run ten frozen methods on 24 paired input conditions: three shapes (512×512, 256×1024, and 512×2048), four unseen spectral families (power-law, log-normal, bimodal, and spectral-gap), and two seeds per shape–family pair. Each base matrix is normalized once in FP64, rounded separately to FP64, FP32, FP16, or BF16, and then shared by all methods at that precision. No coefficients are refitted. Output singular values are measured from a symmetrized FP64 YY^{T}, so the reported error reflects the stated execution precision rather than a low-precision diagnostic eigensolver. The expanded screen includes direct and restarted-Gram realizations of the frozen learned tied control and the four-layer full-restart Variant used in training.

The five-layer restart Variant has the lowest mean MSE in FP64 and FP32 (about 0.05105 versus 0.0725 for Direct PE). In FP16, learned direct is lowest at 0.068902, followed by five-layer Basic at 0.070229; Direct PE gives 0.072358. BF16 separates graph depth and realization more sharply. Direct PE remains lowest at 0.066242, closely followed by learned direct at 0.067980. The four-layer restart Variant remains controlled at 0.087320, whereas the five-layer Basic and Variant rise to 1.080396 and 5.056023, respectively. Restarted Gram realizations reduce the severe no-restart Gram-PE failure, but Gram learned direct has a larger worst case (1.663) than fixed Gram PE with restart (0.458). These observations

Table 6: Cross-precision singular-value MSE on 24 matched synthetic matrices per precision. Entries are means; the last column is the worst BF16 per-matrix MSE. Lower is better; bold marks the lowest mean in each precision column. All 960 outputs are finite, but finite output does not imply numerical stability.

method	FP64	FP32	FP16	BF16	BF16 worst
Direct PE, T5	0.072528	0.072507	0.072358	0.066242	0.234
Gram PE, T5, no restart	0.072528	0.072553	0.076779	86.315351	2055.716
Gram PE, T5, full R3	0.072528	0.072519	0.073901	0.158546	0.458
Learned direct, T5	0.070173	0.070163	0.068902	0.067980	0.233
Gram learned direct, T5, full R3	0.070173	0.070178	0.073142	0.150788	1.663
GMBI Basic, T4, no restart	0.122256	0.122256	0.124415	0.146854	0.346
GMBI Variant, T3, no restart	0.150139	0.150139	0.151568	0.203846	0.357
GMBI Variant, T4, full R3	0.073635	0.073642	0.075019	0.087320	0.424
GMBI Basic, T5, full R3	0.067972	0.067960	0.070229	1.080396	5.449
GMBI Variant, T5, full R3	0.051050	0.051046	0.073344	5.056023	106.177

support precision-specific reporting and the paper's FP16 scope, not a claim of uniform improvement across arithmetic formats. This synthetic screen is distinct from the held-out GPT-2 Medium matrix evaluation in Section 4.1.

F GMBI Algorithm and Variant Details

The variant reuses $H_i = F_{i,1}F_{i,2}$ in both state and accumulator updates. The $d_i H_i^2$ path changes the state-transition map g_i, while $e_i H_i$ changes the output map f_i. Gain limits and full-interval guard samples are necessary because decoupling can produce large off-distribution values. No momentum or second named variant is used in this paper.

For the five-layer full-restart graph, initialize $X = M/(1.02\|M\|_{\mathrm{F}} + 10^{-6})$, $S = XX^{\mathsf{T}}$, and $Q = I$. Apply the basic or Variant layer above. Before layer 3 materialize $X \leftarrow QX$, rebuild $S \leftarrow XX^{\mathsf{T}}$, and reset $Q \leftarrow I$; after layer 5 return QX. The final unused state factors are pruned. In FP16, the principal failure chain is Gram rounding, loss of symmetry or positive semidefiniteness, polynomial amplification, accumulator error, and restart outliers. This motivates fitting the suffix on actual FP16 prefix outputs.

Algorithm 3: One layer of the GMBI variant.

Input: Auxiliary state S, accumulator Q, coefficients for layer i

1 $B \leftarrow S^2$;
2 $F_j \leftarrow a_{i,j}I + b_{i,j}S + c_{i,j}B$, $j = 1, \ldots, 4$;
3 $H \leftarrow F_1F_2$;
4 $Q \leftarrow (F_4 + e_iH)Q$;
5 $S \leftarrow H(F_3 + d_iH)$ if another state is required;

The three-layer variant uses this recurrence without restart. The five-layer version uses the identical layer definition on both sides of the full restart. Calling both methods the same variant emphasizes that restart is a schedule applied to the graph, rather than a separate algebraic architecture.

G Coefficient Selection Protocol

Proposed phasewise recipe. We divide a five-layer full-restart iteration into a pre-restart stage (the prefix, layers 1–2) and a post-restart stage (the suffix, layers 3–5). The stages are fitted sequentially rather than as one unconstrained five-layer problem. The robust-only suffix uses the selected prefix's induced scalar interval. For Muon-aware fitting, we freeze the prefix, execute it on calibration inputs in FP16, materialize Q_2X_0, and rebuild a true Gram matrix. The resulting restart spectra calibrate the suffix; no subsequent joint fine-tuning changes the prefix.

Data and objectives. The robust-only arm uses a fixed scalar grid comprising 4096 log-spaced points on $[10^{-3}, 0.1]$ and 8192 linearly spaced points on $[0.1, 1]$; it contains no recorded GPT-2 matrices. Its prefix and suffix are fitted to the corresponding pre- and post-prefix scalar intervals. The Muon-aware arm uses the same dense interval guard and additionally calibrates on Gaussian and near-low-rank matrices of sizes 1024, 2048, and 4096 (two seeds per family and size), together with ten nonzero GPT-2 Small Muon matrices. Each matrix contributes at most 4096 ordered spectral samples, balanced first across families and then across matrices. Two restart scales, 1 and 1.05, are included when constructing the suffix bank. GPT-2 Medium captures are test-only and are never used for fitting, guards, hyperparameter choice, candidate selection, or early stopping.

For a vector of absolute prefix errors e, the robust and Muon-aware prefix fits use the annealed smooth maximum

$$\mathcal{L}_{\mathrm{pre}} = \tau \left[\log \sum_k \exp(e_k/\tau) - \log |e| \right] + 10^{-3}\mathcal{P}_{4.25}(S) + 10^{-6}\mathcal{P}_{64}(Q) + 10^{-6}\mathcal{P}_{32}(\theta),$$

where $\mathcal{P}_c(Z)$ denotes the mean squared excess $\mathrm{ReLU}(|Z| - c)^2$. The error vector contains all guard-grid points and, in the Muon-aware arm, the observed finite-precision matrix points above 10^{-3}. The temperature is annealed from 0.02 toward 0.002.

For the Muon-aware suffix, let $\mathcal{L}_{\mathrm{emp}}$ be the balanced mean squared error on the realized restart bank. We add 2048 log-spaced tail guards on $[10^{-8}, 0.1]$, 2048 linear upper-spectrum guards on $[0.1, 1]$, and 2048 overshoot guards on $[1, u]$, where u is 1.05 times the largest observed restart-state value, but at least 1.05. The optimized loss is

$$\begin{aligned}\mathcal{L}_{\mathrm{post}} = \mathcal{L}_{\mathrm{emp}} + 0.10\mathcal{L}_{\mathrm{tail}} + 0.35\mathcal{L}_{\mathrm{upper}} + 0.25\mathcal{L}_{\mathrm{over}} + 0.20\mathcal{L}_{\mathrm{worst}} \\ + 10^{-3}\mathcal{P}_{1.25}(S) + 10^{-6}\mathcal{P}_{64}(Q) + 10^{-6}\mathcal{P}_{32}(\theta),\end{aligned}$$

where $\mathcal{L}_{\mathrm{worst}}$ is the mean squared error of the 128 largest upper/overshoot residuals. The robust-only suffix instead applies the same smooth-maximum form as the prefix to the complete interval induced by the selected prefix. Thus "Muon-aware" changes both the observed support and the suffix objective; it is not simply a post-hoc coefficient choice on the Medium test set.

Optimization and selection. Table 7 gives the initial multistart phasewise search schedule. Trial 0 starts from the five-layer Polar Express coefficients (lifted into the GMBI factors where needed); trials 1–2 add deterministic Gaussian perturbations of standard deviation 10^{-3} to active coefficients. Inactive factors immediately before restart or at the final layer remain fixed. Within each prefix trial, the saved checkpoint is chosen lexicographically by measured maximum error and then total loss; the best prefix trial is chosen by the same rule. Suffix checkpoints and trials are selected by their stated suffix loss. A candidate must have finite scalar execution; Muon-aware candidates also require finite FP16 prefix execution and a valid restart bank. Experiments use full-precision coefficients. Tables 8–12 give display copies for the matrix and kernel studies. Training configurations and coefficients are specified by the archived per-run manifests.

For the final Basic Muon-aware cell in Table 1, the selected prefix is frozen and the suffix is chosen by the stated Muon-aware objective from finite candidates. Its calibration bank uses the stated synthetic settings and Small captures, materialized on an RTX 3070 Ti with PyTorch 2.6.0/CUDA 12.4, TF32 disabled, and FP16 reduced-precision reduction enabled. Table 1 evaluation is on H100 with PyTorch 2.7.1/CUDA 12.8 and both reduction shortcuts disabled. The full-precision parameters and their SHA-256 manifest are included in the supplementary archive.

Joint five-layer control. For each tied/Basic and robust-only/Muon-aware cell in Table 1, the control optimizes all five layers at once using exact FP64 scalar singular-value rollouts with full restart before layer 3. It never constructs an FP16-materialized suffix bank. The robust calibration bank contains Gaussian and near-low-rank matrices of orders 1024, 2048, and 4096 with seeds 701 and 702; Muon-aware calibration adds ten nonzero GPT-2 Small updates and balances weights across families and matrices. The common objective combines

Table 7: Initial multistart search schedule for phasewise five-layer fitting. The learning rate is multiplied by 0.2 after 60% and 85% of each stage.

Setting	Pre-restart stage (prefix)	Post-restart stage (suffix)
Optimizer / initial learning rate	Adam / 10^{-3}	Adam / 10^{-3}
Optimization steps / trials	4000 / 3	4000 / 3
Gradient-norm clipping	10	10
Arithmetic for fitting parameters	FP64	FP64
Restart-bank execution	FP16; prefix frozen before bank construction	
Initialization	PE; perturbation $\sigma = 10^{-3}$	PE; perturbation $\sigma = 10^{-3}$

weighted final-output MSE, 0.1 times dense-guard MSE, 0.2 times the mean squared error of the worst 128 guard points, and fixed state, gain, and coefficient penalties. Three starts each receive 8000 Adam steps at initial learning rate 10^{-4}, gradient clipping at 1, and a 0.2 learning-rate multiplier after 60% and 85% of the steps; the lowest recorded complete objective selects the coefficients. The control differs from phasewise fitting in objective, calibration-state construction, and optimization settings. It is a fitted numerical control, not a certified global optimum. GPT-2 Medium remains strictly test-only.

The tied model in either recipe enforces $g_i(s) = sf_i(s)^2$. Basic and Variant GMBI learn separate state and readout factors while sharing S, S^2, and H. Published Direct/Gram Polar Express coefficients bypass this fitting and remain fixed external controls. Each fit records its seeds, full-precision parameters, source hash, environment, and metrics.

H Retrained Coefficients

For each factor, the table gives (a, b, c) in $aI + bS + cS^2$. The variant's d_i, e_i are shown once per layer. These tables document the matrix-study coefficients; the five-layer Basic table specifies the final Muon-aware recipe. Coefficients are printed to six decimal places for readability; the experimental artifacts retain full precision.

Table 8: Retrained coefficients for GMBI, four layers. Blank d_i, e_i cells inherit the value printed on the first row of that layer.

layer i	factor j	$a_{i,j}$	$b_{i,j}$	$c_{i,j}$	d_i	e_i
1	1	7.679309	-20.488585	13.866692		
1	2	0.000296	1.144491	-0.409667		
1	3	7.543228	-20.128316	14.201071		
1	4	7.277012	-19.551789	13.371337		
2	1	3.375440	-2.768682	0.616039		
2	2	-0.010390	1.329256	-0.052330		
2	3	4.016415	-2.601788	0.435181		
2	4	3.053040	-2.263236	0.429110		
3	1	4.050722	-2.662399	0.437135		
3	2	-0.247936	2.129411	-0.489209		
3	3	2.741835	-2.632736	1.106943		
3	4	2.562180	-1.612357	0.263288		
4	1	0.000000	0.000000	0.000000		
4	2	0.000000	0.000000	0.000000		
4	3	0.000000	0.000000	0.000000		
4	4	8.992516	-4.664161	0.715287		

I Dao-AILab Results for Original-Muon GPT-2 Small

This appendix reports the Dao-AILab QuACK half of the paired backend comparison described in Section 4.4. It uses the same six seeds, batches, coefficients, evaluation schedule, and step-5100 endpoint as Table 4; only the backend changes.

Table 9: Retrained coefficients for GMBI variant, three layers. Blank d_i, e_i cells inherit the value printed on the first row of that layer.

layer i	factor j	$a_{i,j}$	$b_{i,j}$	$c_{i,j}$	d_i	e_i
1	1	7.898814	-19.505522	13.023665	-2.001174	-1.502290
1	2	0.003416	2.202661	-1.557253		
1	3	8.652325	-18.744591	12.395373		
1	4	8.663908	-20.210124	13.229768		
2	1	3.720402	-2.986425	0.440022	-0.620595	-0.349836
2	2	-0.312029	1.279374	-0.172024		
2	3	4.189177	-3.132774	0.462968		
2	4	3.885673	-2.800885	0.404604		
3	1	-0.024466	3.246305	-1.881726	0.000000	1.865458
3	2	-1.284453	2.313232	-1.152183		
3	3	0.000000	0.000000	0.000000		
3	4	3.437132	-2.393887	0.727802		

Table 10: Retrained coefficients for the GMBI variant with four layers and a full restart before layer 3. Blank d_i, e_i cells inherit the value printed on the first row of that layer.

layer i	factor j	$a_{i,j}$	$b_{i,j}$	$c_{i,j}$	d_i	e_i
1	1	8.333298	-22.788843	15.798305	-0.485491	-0.566520
1	2	-0.005929	1.466951	-0.384784		
1	3	8.284154	-22.809238	15.787299		
1	4	8.401887	-22.610305	15.802653		
2	1	0.676371	0.467169	-0.165220	0.000000	-0.534361
2	2	0.897370	0.367582	-0.322359		
2	3	0.000000	0.000000	0.000000		
2	4	3.968659	-3.054567	0.592958		
3	1	3.208033	-2.618525	0.575287	-0.624601	-0.510847
3	2	-0.110748	2.192954	-0.425461		
3	3	2.981854	-2.501453	0.585567		
3	4	3.487526	-2.233741	0.486393		
4	1	-0.803194	0.368898	1.013048	0.000000	-1.875637
4	2	-0.700368	0.645846	-0.178610		
4	3	0.000000	0.000000	0.000000		
4	4	2.183126	-1.796421	0.368233		

Table 11: Five-layer full-restart Basic coefficients for the final Muon-aware recipe in Tables 1–3. Blank d_i, e_i cells inherit the first-row value.

layer i	factor j	$a_{i,j}$	$b_{i,j}$	$c_{i,j}$	d_i	e_i
1	1	8.069160	-22.532835	16.099535		
1	2	-0.001916	1.165944	-0.648870		
1	3	8.069284	-22.531475	16.099985		
1	4	8.085383	-22.573142	15.817587		
2	1	0.000000	0.000000	0.000000		
2	2	0.000000	0.000000	0.000000		
2	3	0.000000	0.000000	0.000000		
2	4	3.995079	-2.914920	0.535505		
3	1	3.829949	-2.804045	0.512653		
3	2	-0.005417	1.171961	-0.063052		
3	3	3.832529	-2.797829	0.531160		
3	4	3.877260	-2.768159	0.508955		
4	1	3.237751	-2.443363	0.462876		
4	2	0.045552	1.017257	-0.250178		
4	3	2.992627	-2.450986	0.918293		
4	4	3.302738	-2.298096	0.437830		
5	1	0.000000	0.000000	0.000000		
5	2	0.000000	0.000000	0.000000		
5	3	0.000000	0.000000	0.000000		
5	4	2.165084	-1.742377	0.482697		

Table 12: Retrained coefficients for GMBI variant, five layers with full restart. Blank d_i, e_i cells inherit the value printed on the first row of that layer.

layer i	factor j	$a_{i,j}$	$b_{i,j}$	$c_{i,j}$	d_i	e_i
1	1	8.261453	-22.747484	15.808924	-0.393802	-0.387280
1	2	-0.005554	1.422831	-0.315788		
1	3	8.231278	-22.760652	15.805229		
1	4	8.373611	-22.563368	15.829340		
2	1	-0.690074	-0.493675	0.161426	0.000000	0.568756
2	2	0.735837	0.464879	-0.360525		
2	3	0.000000	0.000000	0.000000		
2	4	3.972202	-3.031035	0.568175		
3	1	3.849234	-2.825792	0.519959	-0.366019	-0.310726
3	2	-0.004532	1.362017	-0.183369		
3	3	3.846147	-2.804373	0.520781		
3	4	3.907129	-2.765772	0.507929		
4	1	3.260149	-2.515045	0.518797	-0.600785	-0.328773
4	2	0.063139	1.588414	-0.363470		
4	3	3.388724	-2.306024	0.548921		
4	4	3.413402	-2.308297	0.465116		
5	1	0.390804	-0.015192	-0.271819	0.000000	0.329248
5	2	-0.420784	0.082976	0.291416		
5	3	0.000000	0.000000	0.000000		
5	4	2.281111	-1.705186	0.558131		

Table 13: Dao-AILab QuACK results for GPT-2 Small under the original per-parameter Muon implementation. Values are mean ± sample standard deviation over six matched seeds. Lower is better; bold marks the lowest mean loss and the lowest mean step time.

method	validation loss at 5100	train s	step ms
Direct Polar Express, 5	3.280700 ± 0.001164	2819.665 ± 8.580	553.960 ± 1.685
Gram Polar Express, 5, full R3	3.280733 ± 0.000989	2826.351 ± 9.437	555.273 ± 1.855
Learned direct, 5	3.280717 ± 0.001336	2829.525 ± 18.894	555.897 ± 3.711
Gram learned direct, 5, full R3	3.281167 ± 0.001084	2831.066 ± 18.650	556.202 ± 3.664
GMBI Basic, 4, no restart	3.289983 ± 0.001348	2818.087 ± 6.700	553.652 ± 1.316
GMBI Variant, 3, no restart	3.283667 ± 0.002613	2812.415 ± 10.257	$\mathbf{552.538 \pm 2.014}$
GMBI Variant, 4, full R3	$\mathbf{3.279667 \pm 0.001050}$	2819.354 ± 9.450	553.898 ± 1.858
GMBI Basic, 5, full R3	3.281800 ± 0.000885	2826.057 ± 10.070	555.217 ± 1.978
GMBI Variant, 5, full R3	3.281183 ± 0.000852	2825.396 ± 7.808	555.087 ± 1.535

The Dao backend retains the accuracy ordering: the four-layer restart Variant has the lowest mean loss, 3.279667, versus 3.280700 for Direct PE, 3.280733 for restarted Gram PE, and 3.280717/3.281167 for learned direct/Gram learned direct; the three-layer Variant is fastest at 552.538 ms per step. Relative to the matched compiled-PyTorch rows in Table 4, Dao increases mean step time by 2.485 ms for Direct PE and by 4.497, 2.502, 2.538, 3.263, 4.463, and 3.830 ms for Gram PE, Basic T4, Variant T3, Variant T4 R3, Basic T5 R3, and Variant T5 R3, respectively. These end-to-end results do not contradict the isolated large-batch speedups in Section 4.3: the per-parameter training path presents smaller, heterogeneous matrices with sequential dispatch and less opportunity to amortize backend overhead.

J Reproducibility and Additional Metrics

The nine-method matrix archive contains 243 matched-family, 1512 synthetic, and 504 GPT-2 Medium method–matrix–precision outputs; two Gram-PE/no-restart FP16 outputs are explicit non-finite failure records. The eight final configurations in Table 1 contribute 216 matched-family, 736 unseen-spectrum, and 224 held-out Medium outputs, all finite. Table 2 contains $9 \times 26 = 234$ finite compiled method–input outputs; the paired compiled/Dao archive contains 468 outputs. The final latency table contains $9 \times 2 \times 6 = 108$ method–backend–shape cells and 154 CUDA-event samples per cell, for 16,632 timed polar calls. The original-Muon GPT-2 Small comparison uses six fixed seeds for nine methods under two backends, giving 108 matched step-5100 runs. A seed controls initialization, data shuffling, batch order, stochastic regularization, and low-level nondeterministic scheduling where applicable.

The matched-family set uses orders 1024, 2048, and 4096, three held-out seeds, and Gaussian, geometric-tail, and near-low-rank spectra. The 92 unseen-spectrum conditions span eight spectral families and ten square or rectangular shapes; all inputs and seeds are fixed, and each matrix is normalized and rounded once to FP16 before the eight frozen methods are evaluated. The reported mean weights matrices equally. The real set contains 28 nonzero pre-polar matrices from 14 GPT-2 Medium capture files at stages 0, 10, 40, and 80. The original nine-method real screen separately uses four nonzero Small and eight Medium attention/MLP updates; its synthetic screen uses 384×1536 matrices and seeds 901 and 907.

For latency, each method–backend cell runs in a fresh process on identical deterministic inputs, receives an independent warm-up, and excludes compilation. CUDA events enclose only the polar call; 14 blocks of 11 synchronized samples yield 154 observations per shape. For training, method order is cyclically rotated, backend order is alternated, validation is recorded every 50 steps from 4750 to 5100, and accumulated training-loop time excludes validation and checkpoint intervals.

J.1 Reference eager implementation

The following code mirrors the equations and makes the full restart explicit. The timed training code uses preallocated buffers and compiled helpers, but it has the same graph semantics.

```
def gmbi_polar(G, layers, variant=False, restart_before=()):
    X = G.to(torch.float16)
    transposed = X.size(-2) > X.size(-1)
    if transposed:
        X = X.mT
    X = X / (1.02 * X.norm() + 1e-6)
    anchor = X.contiguous()
    S = anchor @ anchor.mT
    Q = None

    for i, layer in enumerate(layers, start=1):
        if i in restart_before:
            anchor = Q @ anchor # preserve prefix action
            S = anchor @ anchor.mT # rebuild true Gram state
            Q = None # identity accumulator

        I = torch.eye(S.size(-1), device=S.device,
                      dtype=S.dtype)
        B = S @ S
        def poly(c):
            return c[0] * I + c[1] * S + c[2] * B

        last = i == len(layers)
        if last and not variant:
            # The final Basic readout uses only F4; H is dead.
            factor = poly(layer[3])
        else:
            H = poly(layer[0]) @ poly(layer[1])
            factor = poly(layer[3])
            if variant:
                factor = factor + layer[5] * H
        Q = factor if Q is None else factor @ Q

        immediately_before_restart = i + 1 in restart_before
        if not last and not immediately_before_restart:
            F3 = poly(layer[2])
            S = H @ (F3 + layer[4] * H) if variant else H @ F3

    Y = Q @ anchor
    return Y.mT if transposed else Y
```

J.2 Complete matrix aggregates

The factorial artifacts retain per-case singular-value MSE, maximum error, orthogonality residual, singular-value extrema, finite-status, method order, and diagnostic time. Table 1 uses the eight final tied/Basic configurations from the same-input H100 evaluation identified as 208; full-precision coefficient hashes and case-level records specify the exact mapping. All 92 unseen cases are included. Other frozen matrix and training studies retain their own coefficient manifests. Diagnostic timing is not used as a kernel benchmark.

J.3 Held-out generation and multi-shape H100 benchmark details

The eight unseen families are power law ($\sigma_i \propto i^{-0.72}$), sorted log-normal samples ($\exp(1.15z_i)$, $z_i \sim \mathcal{N}(0,1)$), bimodal spectra (one third in $[0.72, 1]$ and the remainder in $[0.008, 0.045]$), spectral-gap spectra (three fifths in $[0.48, 1]$ and the remainder logarithmically in $[10^{-4}, 10^{-1.7}]$), stretched exponential ($\exp(-9\sqrt{t})$), continuous broken power law ($i^{-0.20}$ to $i^{-1.55}$), a high-valued 1/32 spike block with low bulk, and a modulated square-root soft edge, where t is normalized rank. Every spectrum is Euclidean-normalized and embedded as $X = U\,\mathrm{diag}(\sigma)V^{\mathsf{T}}$ with independently seeded orthogonal factors. Shapes, seeds, and coefficients are frozen before evaluation; exact per-case settings are archived with the raw metrics. The 92-case aggregate is descriptive, not a preregistered primary endpoint.

The final nine-method benchmark uses PyTorch 2.7.1/CUDA 12.8 on an H100 80GB and frozen SHA-256-manifested coefficients. Every compiled/Dao method–shape cell has 154 CUDA-event samples excluding compilation and six warm-ups. All 108 final-table cells are finite, pass exit-code and hash validation, and show three CUDA-Graph launches in three profiler probe calls. Table 2 evaluates all methods on the same 26 frozen inputs. No DRAM-byte, occupancy, achieved-FLOP, or power claim is made without Nsight measurement.

The learned tied replay freezes the five quintic coefficient triples of the Table 1 tied Muon-aware phasewise control (source SHA-256 65e5d74058d0420f...). Learned direct recomputes XX^{T} from the rectangular iterate at every layer; Gram learned direct propagates the equivalent Gram recurrence with full R3. Thus restart is algebraically redundant for the direct realization in exact arithmetic, while the two FP16 executions can differ slightly. The same normalized inputs contribute 27 matched, 92 unseen, and 28 Medium outputs to the Table 1 comparison and 26 compiled plus 26 Dao outputs per realization to Table 2. Coefficients were not refitted. These rows are matrix controls; downstream training comparisons are outside Tables 2 and 3.

J.4 FP64-reference polar-factor fidelity

We replay the frozen Table 1 and Table 2 FP16 implementations without refitting. For each normalized, FP16-rounded input actually supplied to a method, a thin FP64 SVD defines $P = \mathrm{polar}(X)$; the same P is shared by methods on that input. For B matrices with smaller dimension n, we measure

$$E_{\mathrm{polar}} = \frac{\|Y - P\|_{\mathrm{F}}}{\sqrt{Bn}}, \qquad \theta = \arccos \frac{\langle Y, P\rangle}{\|Y\|_{\mathrm{F}}\|P\|_{\mathrm{F}}}, \qquad E_{\mathrm{orth}} = \frac{\|YY^{\mathsf{T}} - I\|_{\mathrm{F}}}{\sqrt{Bn}},$$

with tall matrices transposed for the orthogonality calculation and batch Frobenius norms taken over all B matrices. The whole-output angle θ detects a changed output direction but also reflects unequal singular-value scaling; it is not a per-singular-vector or subspace principal angle. E_{orth} is a spectral diagnostic and cannot detect a pure rotation. We give mean and 95th-percentile E_{polar} within each group, plus all-case maximum E_{polar} and mean θ (degrees) and E_{orth}. Each mean weights the evaluated matrices equally, as in the original tables. No evaluated input fell below the preregistered relative 10^{-8} near-rank-deficiency threshold; polar directions at exact rank deficiency would require a separate convention.

Table 14: Table 1 methods replayed against the FP64 polar factor on identical FP16 inputs. Subset columns report mean / 95th-percentile normalized polar error within the named subset. The rightmost three columns use all 147 inputs per method: maximum polar error, mean whole-output angle in degrees, and mean orthogonality residual. Lower is better.

	Subset E_{polar}: mean / P95			All 147 inputs		
method	matched (27)	unseen (92)	Medium (28)	max E_{polar}	mean θ (°)	mean E_{orth}
tied robust joint	.5061/.8539	.2113/.5924	.2902/.7947	.8539	15.56	.3808
Basic robust joint	.4961/.8480	.1998/.5855	.2687/.7881	.8480	14.77	.3550
tied aware joint	.5018/.8468	.2148/.5869	.2753/.7877	.8468	15.41	.3746
Basic aware joint	.4891/.8315	.2067/.5724	.2691/.7721	.8315	14.86	.3665
tied robust phasewise	.5001/.8464	.2100/.5854	.2693/.7868	.8464	15.19	.3651
Basic robust phasewise	.4962/.8428	.2034/.5822	.2627/.7846	.8428	14.81	.3535
tied aware phasewise	.5035/.8511	.2119/.5894	.2760/.7915	.8511	15.36	.3695
Basic aware phasewise (final)	.4633/.7945	.2136/.5475	.2530/.7323	.7945	14.23	.3906

The final Basic phasewise recipe retains the lowest matched and Medium mean polar error and the smallest all-case maximum. Its unseen mean is above the robust-only Basic joint and

phasewise controls, while its unseen 95th percentile is lower. Thus the singular-value-MSE ranking alone does not describe all aspects of finite-precision polar fidelity.

Table 15: All nine Table 2 methods replayed against the FP64 polar factor. Subset columns report mean / 95th-percentile normalized polar error within the named subset. The rightmost three columns use all 26 nonzero inputs per method: maximum polar error, mean whole-output angle in degrees, and mean orthogonality residual. The three exactly zero historical Small captures are excluded, as in Table 2. Lower is better.

	Subset E_{polar}: mean / P95					All 26 nonzero inputs		
method	Small (4)	Medium (8)	Gaussian (2)	tail/low rank (4)	unseen (8)	max E_{polar}	mean θ (°)	mean E_{orth}
Direct PE	.3899/.5133	.4254/.7781	.0932/.0934	.7869/.9396	.2031/.5058	.9396	23.29	.4808
Gram PE R3	.3900/.5133	.4254/.7779	.0931/.0932	.7845/.9367	.2040/.5047	.9368	23.24	.4816
Learned direct T5	.3866/.5112	.4211/.7767	.0806/.0807	.7860/.9392	.1918/.5045	.9392	22.87	.4681
Gram learned direct T5 R3	.3867/.5112	.4211/.7765	.0805/.0806	.7836/.9364	.1926/.5032	.9364	22.82	.4688
Basic T4 NR	.6039/.7520	.6455/.9112	.0266/.0266	.8343/.9588	.2923/.5781	.9588	31.69	.6091
Variant T3 NR	.6548/.7975	.6973/.9298	.0817/.0819	.8451/.9614	.3453/.5936	.9614	34.42	.6640
Variant T4 R3	.4249/.5592	.4621/.8108	.0405/.0406	.7944/.9429	.1891/.5167	.9429	23.89	.4782
Basic T5 R3	.3556/.4569	.3925/.7159	.0592/.0593	.7667/.9236	.1979/.4788	.9236	21.80	.4902
Variant T5 R3	.2787/.3745	.2957/.6394	.0173/.0174	.7397/.9030	.1370/.4462	.9030	17.53	.3597

The five-layer Variant has the lowest mean E_{polar} in every Table 2 group, and the lowest overall 95th percentile (0.8437 versus 0.9028 for Direct PE). The near-low-rank worst case still reaches 0.9030, which limits any claim of uniform approximation. An independent H100 replay reproduces the Table 1 singular-value means within 4.90×10^{-7}. The supplementary archive contains case-level metrics, input identifiers, coefficient and source hashes, exit codes, and completion markers for every reported entry.

K Paired GPT-2 Small Training Analysis

We define each paired difference as four-layer full-restart Variant minus the baseline on the same seed and backend; negative values favor the Variant. The independent units are the six seeds. Checkpoints at steps 4750, 4800, 4850, 4900, 4950, 5000, 5050, and 5100 reuse the fixed validation stream and are correlated. For the late-window statistic, we first average the eight differences within each seed and then form a paired t interval across the six seed means (five degrees of freedom). All intervals below are nominal two-sided 95% intervals. Losses are logged to four decimal places.

Table 16: Individual compiled-PyTorch paired differences at step 5100. Time is the logged mean training-step latency in ms.

	Loss difference		Step-time difference (ms)	
Seed	vs. Direct PE	vs. Gram PE	vs. Direct PE	vs. Gram PE
842641	-0.0006	+0.0001	-2.14	-1.47
193171	-0.0025	-0.0020	+0.39	-0.26
284759	-0.0003	-0.0012	-1.02	+0.66
376127	-0.0007	-0.0004	-0.57	+0.67
468593	-0.0023	-0.0013	-1.05	-0.25
759431	-0.0013	-0.0020	-0.65	-0.20

Table 17: Paired mean differences [nominal 95% CI]. Loss columns are in units of 10^{-3}; the time column is in ms. The late-window loss uses one eight-checkpoint average per seed.

Backend / baseline	Final loss $\times 10^3$	Late-window loss $\times 10^3$	Step time (ms)
compile / Direct PE	-1.283 [-2.255,-0.311]	-1.329 [-2.241,-0.417]	-0.840 [-1.704,0.024]
compile / Gram PE R3	-1.133 [-2.023,-0.244]	-1.154 [-2.002,-0.307]	-0.142 [-0.967,0.684]
dao / Direct PE	-1.033 [-2.390,0.323]	-1.085 [-2.405,0.234]	-0.062 [-1.763,1.640]
dao / Gram PE R3	-1.067 [-2.584,0.451]	-1.098 [-2.634,0.438]	-1.375 [-2.027,-0.723]

Under compiled PyTorch, all 48 seed–checkpoint differences favor the Variant over Direct PE; checkpoint-mean improvements range from 0.001283 to 0.001367. Against Gram PE,

41 differences favor the Variant, four tie at logging precision, and three favor Gram PE. The Direct-relative endpoint gain corresponds to approximately 0.128% lower perplexity. This is a small absolute gain whose direction persists throughout the observed late-training window.

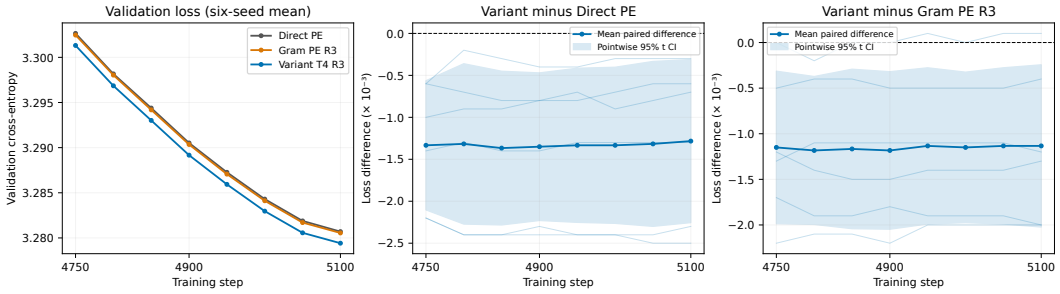

Figure 1: Compiled-PyTorch late-training validation curves and paired differences. Thin lines show individual seeds; shaded bands are pointwise nominal 95% paired t intervals, not simultaneous bands. The 48 plotted pairs are six correlated trajectories.

The optimization objective naturally defines the directional alternative $H_1 : \mu_d < 0$ against $H_0 : \mu_d \geq 0$. This supplementary seed-level analysis is post hoc and provides supportive rather than confirmatory evidence. Adjusting the five one-sided GMBI-versus-baseline tests separately for each fixed PE baseline gives the four-layer full-restart Variant Holm-adjusted paired-t $p = 0.0484$ against Direct PE and $p = 0.0552$ against Gram PE. Learned-tied methods are diagnostic controls, not configuration-selection candidates. The loss reduction relative to Direct PE is significant at the 0.05 level after Holm adjustment over the five GMBI candidates ($p = 0.0484$); the corresponding adjusted p-value against Gram PE is 0.0552. Together with the paired differences and nominal two-sided confidence intervals, these results support a small, persistent loss improvement in the evaluated training setting.

The supplementary archive contains the per-seed logs, source hashes, and paired statistics used by Table 4.